Computers in Human Behavior, Vol. 6, pp. 51-68, 1990
Printed in the U.S.A.

0747-5632/90 $3.00 + .00

An Experimental Evaluation of Different Amounts of Receptive and Exploratory Learning in a Tutoring System

Franz Schmalhofer and Otto Kühn

German Research Center for Artificial Intelligence

Paula Messamer

University of Colorado

Rhona Charron

McGill University

Abstract—*A general model of knowledge acquisition is presented which distinguishes receptive and exploratory learning as two separate learning modes. In addition, learning materials are differentiated into situational examples and text information. A learner's prior knowledge is represented as heuristic rules and domain specific knowledge. The different learning methods, which may be arbitrarily combined, are implemented as PROLOG meta-interpreters so that the costs and benefits of the different learning methods can be assessed. For the first hour of learning the programming language LISP, a tutoring system was developed in which the amount of exploratory and receptive learning can be manipulated. In an experimental study with three different learning conditions, learning in a basic exploration environment (with no tutor) was compared to learning with a selective tutor and learning with a constant tutor. The results showed that the selective tutor condition was most effective: In this condition the students required the least amount of time for acquiring knowledge and solving the criterion test tasks, while solving an equal number of programming tasks correctly. The results thus show that the proposed tutoring strategy was quite successful in combining the advantages of learning by exploration and learning from instructions.*

Intelligent tutoring systems provide a new means of combining learning by exploration and learning from instructions (receptive learning) in such a way that the particular combination utilizes the advantages of both learning methods, while their disadvantages are avoided. In addition, the exploration and instruction sequences

Requests for reprints should be addressed to Dr. Franz Schmalhofer, German Research Center for Artificial Intelligence, Postfach 2080, D-6750 Kaiserslauten, West Germany. e-mail: schmalho, informatik. uni-kL.de

can be tailored to the individual needs of a learner. Since the teaching strategy of a tutoring system can be made more explorative or more instruction-based (Dede, 1986), an adequate mixture of receptive and exploratory learning must be determined for every tutoring system. In researching this question, we will first describe the different learning methods and their possible combinations. An integrative model of these methods of human knowledge acquisition will be outlined. This model has been more completely described by Schmalhofer (1986), and Schmalhofer, Kühn, and Messamer (1989).

Although the proposed model is general, we will restrict its presentation to the learning of the programming language LISP. Since only relatively simple LISP concepts (possible types of inputs to the LISP interpreter, elementary LISP functions like FIRST, REST, etc.) will be considered, the reader is assumed to be knowledgeable about the domain. In any case, the reader may obtain the necessary understanding of these functions from the examples presented in the paper. The model is then used to discuss the strengths and weaknesses of the different learning methods and a tutoring system for learning basic LISP is developed. In the tutoring system the degree of receptive and exploratory learning can be manipulated for any given learning history. This is accomplished by simply changing two parameters (n and m) in the student model (i.e., a simple version of the cognitive model) of the tutoring system. The effectiveness of three different levels of exploratory learning is then evaluated by having all learners solve a set of programming tasks after they had been tutored under the three conditions.

1. RECEPTIVE AND EXPLORATORY LEARNING

All learning can be described as receptive or exploratory. In receptive learning a teacher or tutor presents some information so that the learner can acquire new knowledge. In exploratory learning, on the other hand, the learners themselves can choose which additional knowledge they want to acquire by instigating an appropriate interaction which yields some informative result.

Receptive learning can be further divided according to the type of material. The tutor may present general statements in the form of a text. This learning method has been called "learning by being told" and "advice taking" (Mostow, 1983). The tutor may also present concrete examples of a situation which can be seen as specific solutions to some particular problem. The student can thus learn from a demonstration consisting of a sequence of examples. This learning method has been more generally called "learning from examples" (Winston, 1975).

Exploratory learning can be similarly divided according to type of material. The type of material available to a learner depends upon what is being explored. If an environment is explored only concrete situational examples can be obtained. For example, when a learner generates some input to a LISP-interpreter (environment) his input and the response of the environment constitute a situational example. This method has been called "learning by discovery." If some tutor or teacher is available for interrogation by the student much more general questions can be asked and answered. The learning material may therefore also consist of text sentences. This method is called the question and answer method. An overview of this classification scheme is shown in Table 1.

For all described learning methods a learner will utilize his prior knowledge when

Table 1. Classification of learning methods

	Responsibility for Teaching Strategy	
Learning Material	tutor (receptive)	student (exploratory)
Situational Examples	demonstration	discovery learning
Text	learning by being told	question and answer method

generating or encoding new learning material. This prior knowledge consists of general heuristics (Lewis, 1988) and domain specific knowledge. Depending upon the particular learning method, different kinds of prior knowledge are utilized (Schmalhofer & Boschert, 1988). The costs and benefits of using a particular learning method also depend on the available prior knowledge. As knowledge is acquired with one learning method, it may become beneficial to compensate for the shortcomings of this method by switching to a different learning method. Therefore a combination of these methods should be used. All learning can be described as a sequence of episodes of variable lengths where each episode involves exactly one learning method. For each learning method the goals of knowledge acquisition, the learner's prior knowledge, and the resulting information processes should be specified. By combining the different learning methods an integrative model of knowledge acquisition is then obtained.

2. AN INTEGRATIVE MODEL OF KNOWLEDGE ACQUISITION

Processing Goals of Knowledge Acquisition

The purpose of knowledge acquisition is to form and/or update a mental or situation model (van Dijk & Kintsch, 1983) so that it will be well suited for solving one or several target tasks. Normally at the time of knowledge acquisition the tasks which must be performed in the future are not yet completely known. Nevertheless, the goal of knowledge acquisition is to construct the situation model so that various target tasks can be solved relatively easily. Knowledge acquisition can thus be seen as a problem-solving process in which partial solutions for several yet unknown tasks are to be constructed. If possible, the partial solutions should be formed so that little additional processing is required for performing the task: An operationality criterion should be satisfied in the sense that the acquired knowledge is operational for performing those tasks (Mitchell, Keller, & Kedar-Cabelli, 1986) which will eventually arise. In other words, the goal in developing the situation model is to provide operational knowledge which is also general enough to be applied to relatively new situations. This processing goal will be called the knowledge acquisition or KA-goal.

In receptive as well as in exploratory learning, text materials usually present general information. In order for this information to be useful (i.e., to be operational for solving specific tasks), the KA-goal requires that more specific information be derived from it. The general text statements must therefore be transformed into more task-specific knowledge. Situational examples, on the other hand, present very specific information. And, in order for an example to be useful for solving

other tasks, the KA-goal requires that more general information be inferred from it. Examples must be generalized so that the resulting knowledge is applicable for a larger variety of tasks. Thus two different subgoals, one for learning from text and one for learning from examples, result from the general KA-goal. Ideally, the general KA-goal should thus yield a (coherent) situation model which contains knowledge that is operational and at the same time general enough to be applied to a number of tasks.

Whereas in receptive learning the learning material is provided by the tutor, in exploratory learning the student must initiate some informative interaction with the environment or explicitly question the tutor. Knowledge acquisition in exploratory learning is therefore always highly individualized (Kühn & Schmalhofer, 1987). For receptive learning a similarly high degree of individualization can hardly be obtained in a tutoring system. Since learning can only be explained on the basis of some existing prior knowledge, the representation of knowledge will be discussed before the different learning methods are described.

Representation of Knowledge:

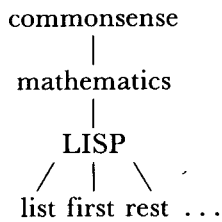
Three types of knowledge are distinguished: (a) known rules and facts, (b) hypotheses, and (c) heuristics which can generate hypotheses. This knowledge is represented by PROLOG clauses in a knowledge base in the three following forms:

1. known(Domain,FactOrRule,Info)
2. hyp(Domain,FactOrRule,Info)
3. heur(Domain,FactOrRule,Info).

Rules and facts which are known to be true are represented by form (1). Hypotheses which were inductively formed and may therefore be incorrect are represented by form (2). Form (3) is used to specify heuristics which are applied for generating and modifying hypotheses. Please note that in PROLOG capital letters at the beginning of a word identify variables.

In order to facilitate the access of the relevant knowledge each clause is assigned to a Domain so that the knowledge of the relevant domains may be selectively accessed. FactOrRule specifies the knowledge contents. Info provides additional information about a knowledge element such as confidence and usefulness counters, which are initialized to (1,1).

Domains are hierarchically organized. For the learning of elementary LISP the relevant knowledge is structured according to the following hierarchy:



Commonsense knowledge includes, for example, the transitivity of subclass relations, counting, and general heuristics. When learning LISP, commonsense knowl-

edge is accessed if the subordinate more specific knowledge is insufficient for accomplishing the KA-goal.

To the domain of mathematics belongs the knowledge about functions in general (i.e., function schema with the slots: number of arguments, type of arguments, and input/output relation). This function schema is represented by the following rule:

```
correct_function(Fname,Arguments,Result) :-
  correct(number_of_arguments,Fname,Arguments),
  correct(type_of_arguments,Fname,Arguments),
  correct(io_relation,Fname,(Arguments,Result)).
```

The LISP knowledge consists of the definitions of atoms and lists and of how the LISP-interpreter evaluates a system input. The following rule is an example from this knowledge domain:

```
eval(Input,Result) :-
      funcall(Input,Fname,ArgSpecs),
      apply(eval,ArgSpecs,Arguments),
      correct_function(Fname,Arguments,Result).
```

The three described knowledge domains (Commonsense, Mathematics, and LISP) are assumed to form the learner's prior knowledge. With this prior knowledge the specific functions LIST, FIRST, REST, SET, etc. are to be learned with the different learning methods.

Learning by Being Told

Based upon the described prior knowledge the function FIRST can be learned from the following sentences: "FIRST is a LISP function. It takes exactly one argument. The argument must be a list. The function FIRST returns the first element of the argument." These sentences can be easily transcribed into PROLOG-clauses. This transcription is performed manually and results in:

```
lisp_function(first).
required(number_of_arguments,first,1).
required(type_of_arguments,first,list).
required(io_relation,first,the_first_element_of_argument).
```

How further knowledge is acquired from these clauses is specified by a PROLOG meta-interpreter which uses mainly forward inferences to achieve the KA-goal. This meta-interpreter uses the prior knowledge, already stored in the knowledge base, to derive more specific knowledge from each statement. Figure 1 shows the main processing steps in the form of a flow chart. Each statement is first stored in the knowledge base. The meta-interpreter then attempts to derive a forward inference from this statement and the prior knowledge and then determines whether this inference is useful. If the inference is useless another inference will

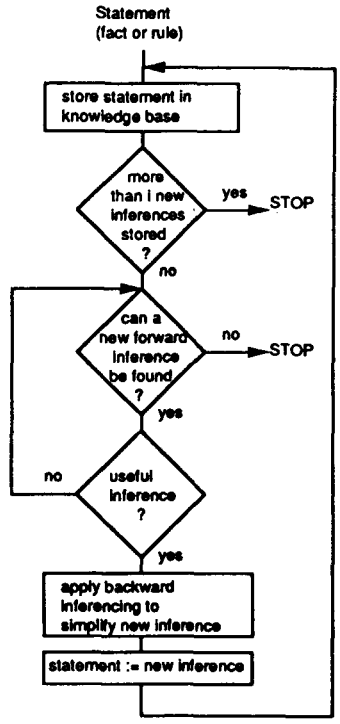


Figure 1. Flowchart diagram of learning by being told.

be derived. If the inference is found useful then it is simplified by applying backward inferencing. A more operational rule is thus generated by dropping those conditions of the rule which unify with the newly presented fact. The variable bindings established in the unification are maintained. The so obtained inference can then be used to derive further inferences.

When generating inferences the knowledge base is searched from the specific to more general Domains. Since the most recently acquired knowledge is always stored at the top of each Domain, recently acquired knowledge is preferably utilized. A large (possibly infinite) number of rules may be derived from each statement where only a small number will be useful. Therefore, this rule construction process has been restricted. Only a limited number of inferences are obtained ($i = 1 \ldots 4$), and only the first useful inference at each level is stored. This inference restriction proved to be quite successful. Whenever an inferred rule is used during further inference processing the usefulness counter stored in INFO is incremented by 1.

For example, after two processing steps the following more operational rule is obtained from the statement "lisp_function(first)":

```
eval(Input,Result) :-
        Input = [first | Argspecs].
        apply(eval,Argspecs,Arguments),
        correct_function(first,Arguments,Result).
```

This rule can be seen as a more specialized rule for coding a function call with the function first. In particular, it specifies that the form of the input is: "(first Arg-specs)," with additional constraints being satisfied. When more facts of the respective LISP functions are learned this rule becomes further operationalized. In summary, by specializing text statements these processing steps yield partial solutions to some potential tasks. The flowchart of Figure 1 shows the different processing steps in learning by being told.

Learning from Examples

Alternatively, LISP functions like FIRST or LIST can also be learned from examples such as:

(LIST 'A 'B) → (A B)
(LIST '(A B) 'C '(D E))→((A B) C (D E)).

In PROLOG notation these examples are represented by:

interaction([list,$a,$b,],[a,b]).
interaction([list,$[a,b],$c,$[d,e]],[[a,b],c,[d,e]]).

How additional knowledge is acquired from these examples is again specified by a PROLOG meta-interpreter. This meta-interpreter uses the prior knowledge, already stored in the knowledge base, to explain on the basis of this more general knowledge that the example is correct. Initially, only the rules and facts of type known are allowed to explain the correctness of the example. If the example contains truly new information (i.e., facts or rules which cannot be derived from the clauses of type known), no such explanation can be constructed. In this case, the heuristic rules are employed to generate hypotheses so that an explanation of the example can be found. These hypotheses thus fill the gaps in the knowledge required for explaining the example (Hall, 1988). These generated hypotheses represent the new knowledge which was acquired from the example and the prior knowledge. Additional and operational knowledge is constructed by using the constructed explanation to perform an explanation-based generalization of the example (Mitchell, Keller, & Kedar-Cabelli, 1986). If the explanation- based generalization was derived from the constructed explanation (technically speaking a proof tree) which contains hypotheses, the hypotheses are stored as part of the derived operational specialization.

In order to construct an explanation based generalization of the example (i.e., an operational specialization of the concept) an operationality criterion must be specified. A low criterion (i.e., only the concepts in or near the leaves of the proof tree are considered operational) yields highly specific specializations whereas a higher criterion yields more versatile specializations. In an early learning phase when there is little prior domain knowledge the operationality criterion is assumed to be low. Consequently, highly specific operational knowledge is acquired at the beginning.

Later on in learning, each newly constructed specialization can be compared to the already existing specializations of the same general concept. If a generalization of the new and the old specialization can be obtained by moderately increasing the operationality criterion, a more general specialization is constructed and stored.

For learning from examples a detailed description of the sequence of processing steps is shown in the flow chart of Figure 2.

During the processing of the first example (LIST 'A 'B)→(A B) the following four hypotheses are generated:

1. hyp(list,lisp_function(list),(1,1)).
2. hyp(list,required(number_of_arguments,list,2),(1,1)).
3. hyp(list,required(type_of_arguments,list,[atom,atom]),(1,1)).
4. hyp(list,required(io_relation,list,list_of_arguments),(1,1)).

The confidence in the correctness of these hypotheses and their usefulness is indicated by the Info (1,1). When an hypothesis is reused INFO is updated as described by Schmalhofer (1986). For the second example "(LIST '(A B) 'C '(D E)) → ((A

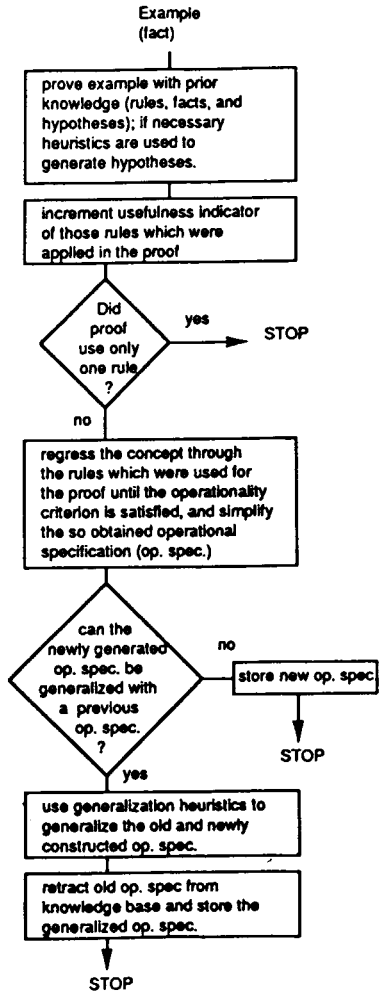


Figure 2. Flowchart diagram of learning from examples.

B) C (D E))" no explanation can be directly constructed from the generated hypotheses. Therefore generalization heuristics are applied. And the hypotheses 2 and 3 become modified to:

2*. hyp(list,required(number_of_arguments,list,at_least(2)),(2,1)).
3*. hyp(list,required(type_of_arguments,list,s_expr),(2,1)).

Together with this generalization of hypotheses, the operational specializations which depend upon these hypotheses must also be generalized. In this case generalization heuristics (inductive learning mechanisms) are used to obtain a generalization of the two more specific explanations (proof trees). Generalization heuristics and inductive learning mechanisms are thus only used when there is insufficient domain knowledge for explanation based learning (Hall, 1988). Our model thus provides an integration of similarity and explanation based learning. Any inductive bias is explicitly represented by heuristic rules (compare Lewis, 1988). Since similarity based learning is only used where explanation based learning fails, all available domain knowledge (facts and previously generated hypotheses) are used whenever possible and new hypotheses are generated parsimoniously.

Learning by Exploration

In learning by exploration the learner himself must instigate the learning process by generating an input to an environment or by asking a teacher a question. Whereas a teacher can basically answer all questions, general and specific, an environment will only respond to very specific types of probes (e.g., the only type of questions a LISP-interpreter can answer are "What is the result of evaluating a particular input?").

A question and answer to that question constitute an interaction with the teacher or the environment which defines a general statement or an example. From such a statement or example knowledge can only be acquired if the interaction is informative (i.e., it helps to fill knowledge gaps). This knowledge acquisition is assumed to proceed according to the specifications of Figures 1 and 2. For modeling learning by exploration, therefore only the generation of inputs and questions must be explained.

A flowchart of this process is shown in Figure 3. The knowledge base is first searched for gaps and insufficiently tested hypothesis or some insufficiently operationalized fact or rule. Insufficiently operationalized knowledge can for example be identified from the lack of an operational specialization in the particular domain. From the selected knowledge item a question can then be generated by forward and backward inferencing. For exploring the LISP system the set of answerable questions is determined by the goal:

interaction(Input,Result),ground_term(Input).

A hypothesis to be tested may be:

required(number_of_arguments,list,2).

From this the question

interaction([list,$ a, $ b], Result)

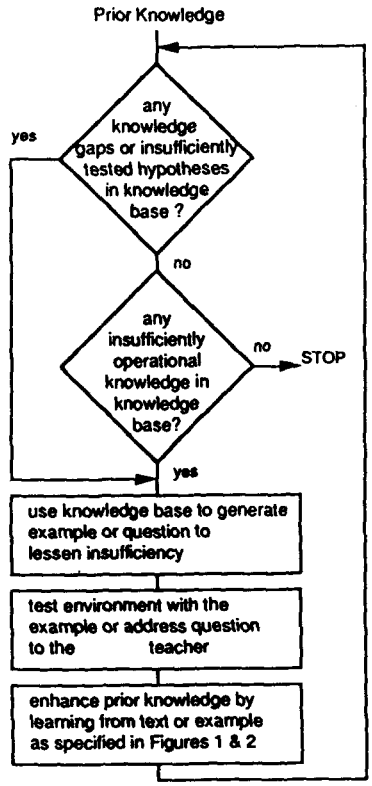


Figure 3. Flowchart diagram of learning by exploration.

is obtained, which specifies a valid input to the LISP interpreter and is consequently answered by the LISP-interpreter.

A hypothesis can also be tested by trying to falsify it. In this case an alternative hypothesis, for instance "required(number_of_arguments, list,3)" is generated. By deriving examples with the modified hypotheses, positive examples allow generalization of the tested hypotheses and negative examples require a specialization as described in Schmalhofer (1986).

Comparison of Learning Methods

The described simulation can now be used to reveal the strengths and weaknesses of the various learning methods as a function of the learner's prior knowledge. Learning by exploration has the advantage that the student himself can determine what to learn. Thereby individual knowledge gaps can be filled and a learner can maintain his individual knowledge organization. (In learning from instructions a student must at least temporarily adapt to the tutor's organizational structure.) As long as a learner has sufficient prior domain knowledge for asking relevant questions or generating informative interactions, learning by exploration will thus be very effective.

The effectiveness of learning by exploration depends, however, more severely upon the learner's prior knowledge than learning from instructions. Contrary to learning from instructions, learning by exploration requires the generation of relevant hypotheses. Hypotheses are always generated on the basis of already existing knowledge (Schmalhofer & Kühn, 1988). If there is insufficient knowledge for generating hypothesis, the success of learning by exploration will stall. Therefore, learning by exploration is most useful, when a learner has a definite agenda about what knowledge is to be acquired. Such an agenda may for example be given by a list of concepts to be learned and a cognitive schema (e.g., function schema), whose slots have to be filled to define each of the target concepts.

In learning by being told novices can usually only generate a small number of useful inferences because of their limited prior knowledge. In this learning mode novices will therefore predominantly store the explicitly stated information whereas advanced learners may in addition successfully apply inferencing and may thereby construct a coherent situation model of the domain (Schmalhofer, 1982).

Novices may apply inferencing when learning from examples by applying their general heuristic knowledge. The resulting hypotheses may, however, be incorrect. What and how many incorrect hypotheses are generated depends upon the consistency between the learner's common sense knowledge and the domain about which knowledge is to be acquired. Advanced learners, on the other hand, may use examples to operationalize their general knowledge for more specific situations without learning anything qualtitatively new.

In addition, different sequences of materials (e.g., text before examples vs. examples before text) can be assessed. If examples are studied after text (all the relevant rules and facts have been told) the learner possesses more (general) domain knowledge and may apply these rules and facts to construct operational specializations from the presented examples. Through the previously derived inferences less reasoning is required. Studying and inferencing from text thus facilitates the formation of operational knowledge from subsequent examples.

Quite different processing requirements arise for text which is processed after (corresponding) examples have already been studied. In this case, hypotheses may have already been generated from the examples. In addition, operational specializations may have been constructed. It is therefore possible that the text will contradict the already generated hypotheses. These incorrect hypotheses must consequently be retracted from the knowledge base (or appropriately modified). In order to maintain a consistent knowledge base, all inferences derived with incorrect hypotheses must also be retracted. On the other hand, if the text confirms the generated hypotheses, they can be asserted to be of type known, in which case nothing qualitatively new is learned.

In summary, it is obvious that prior domain knowledge will facilitate all the different types of learning and that learning by exploration is the most desirable learning method, provided that:

1. it can be carried out successfully (i.e., the finally resulting knowledge is correct) and
2. the time required for successful learning is acceptable.

These conditions will be generally met, when the student's activated knowledge (heuristic as well as domain knowledge) provides enough constraints, so that the

search space for generating interactions with the environment is of a reasonable size and encompasses the correct hypotheses. In this case the hypothesis-generation process and learning by exploration will be successful. Self-generated hypotheses will be remembered better than the information which is supplied to a learner (Slamecka & Katsaiti, 1987). Hypothesis generation (when learning from examples and learning from exploration) is consequently more effective than learning from text whenever the search space for generating hypotheses is sufficiently restricted. The restriction of the search space is determined by the structure and amount of the learner's domain knowledge. Whenever a learner has insufficient domain knowledge for constructing approximately correct hypotheses, appropriate receptive learning episodes should be administered in a tutoring system. These receptive learning episodes will further constrain the search space for generating hypotheses, so that learning by exploration may again become successful. This is the fundamental idea of the tutoring system described in the next section.

3. A TUTORING SYSTEM FOR LEARNING ELEMENTARY LISP

At first the learners' heuristic knowledge is assessed and they are instructed about the most basic facts of the domain. All learners then begin to explore the learning domain (LISP system), while their exploration behavior is monitored. All interactions of the learner with the system are interpreted in terms of the described cognitive model (learning from examples). Thereby it can be determined whether or not a learner can still acquire new knowledge through his explorations. When a student encounters an impasse in his exploration behavior, for which the cognitive model determines that he cannot overcome this impasse himself, an appropriate instruction is presented (example or text information). Text information is used to direct a student's attention to search spaces, which contain relevant hypotheses. From syntactically incorrect inputs, examples are constructed on line, so that they are suited for testing those hypotheses, which the student supposedly intended to test. After such tutor instructions learning by exploration should consequently become successful again. The advantages of learning by exploration can thus be utilized while its disadvantages are avoided.

In order to test the adequacy of the proposed model for intelligent tutoring, a learning environment with diagnosis and teaching components was developed. This learning environment was restricted to the first few hours of learning. This tutoring system is applied in the following way:

1. A learner's relevant knowledge is assessed and entered into the learner's individual knowledge base.
2. Learning by being told is used to induce additional prerequisite knowledge, needed for making learning by exploration successful.
3. An exploration agenda is provided to the learner (e.g., learn the functions FIRST, REST, SET, and LIST). A simple correct example is shown for each function. (In other applications such as text editing a learner may already have an agenda himself.)
4. The student's explorations are then monitored by modelling the knowledge acquisition process as previously described (i.e., learning from examples).

5. The resulting student model which is being updated on-line with every exploration episode is used to determine when learning from a specific example and learning by being told should be applied: Learning from an example is applied when a user needs help with specific details. Learning by being told is applied to direct a learner's attention and provide more global conceptual information.

In the tutoring system the amount of receptive learning can be manipulated in the following way: Examples and advice can be presented whenever a student behaves clearly nonoptimal. Alternatively, learners may be allowed to learn from their own mistakes. Here, examples and advice will only be presented after a learner has also failed to learn from his own mistake. We will now describe the various components of the tutoring system.

The Basic Exploration Environment

By acting in this learning environment the student can learn within an hour:

- the correct syntax for an input to the LISP system
- how a given input is evaluated and what result is returned
- the number and type of arguments which a function requires
- that quoted expressions, bound atoms or function calls can be specified as arguments.

At the beginning of a learning session only the names of the functions which the learner can explore are shown on the top of the screen. The learner must then generate an input to the LISP system. In order to avoid unnecessary typing errors, only characters that are valid in LISP (letters, digits, blank, parentheses, and the quote) can be typed, and only lines with balanced parentheses are accepted as inputs. In addition, colors are used to indicate the level of nesting in the expressions. As these features help to generate syntactically correct training examples, they should reduce the number of trivial and useless syntax errors.

The generated inputs are then evaluated by the LISP-interpreter and the result of the evaluation or an error message is returned. The diagnosis component, interprets the learning progress in terms of the described model and detects which information is needed to redirect a learner's attention or to further restrict the search space used for generating hypotheses.

Diagnosis Component

The diagnosis component is driven by a greatly simplified version of the described cognitive model. In particular only the templates are stored while the proof trees are discarded. For a sequence of input examples, templates are thus constructed and modified. A simple template is constructed from the first example. How this template is modified and a separate template constructed from the fourth example is shown below. (Note: For reasons of simplicity a LISP-like notation is used to denote the various operational specializations, also called templates. Also, ?A denotes a single member and +A an arbitrary number of members of a class.)

input sequence:

1. (FIRST '(A B))
2. (FIRST '(X (Y Z)))
3. (FIRST '((G H)))
4. (FIRST FRIENDS)

constructed templates and modifications:

1. (FIRST '(A B)), (A is-atom), (B is-atom)
2. (FIRST '(?A ?B)), (?A is-atom), (?B is expr)
3. (FIRST '(?A +B)), (?A is-expr), (+B is-expr)
4. (FIRST ?A), (?A is-bound-atom)
   (FIRST '(?A +B)), (?A is-expr), (+B is-expr).

Teaching Strategy

Assistance in learning by exploration is provided on two occasions: (a) when a sequence of n inputs which are redundant in terms of the constructed templates has been detected, or (b) when a sequence of m errors has been detected. Since redundant examples are usually generated when a learner does not know how to proceed, rather general help information is provided in the form of a text. For instance, for

example 1: (FIRST '(A B))
example 2: (FIRST '(X Y))

the help information is:

The argument for the function FIRST can be a list of any complexity. And for

example 1: (FIRST '((A B) (C D)))
example 2: (FIRST '(X Y))

the help information is:

The argument for the function FIRST can also be bound atom or a function call.

Such help information should assist the learner for opening a new search space, in which new informative examples can be constructed. When a sequence of errors is detected, the learner presumably wanted to perform a task, but failed to construct a completely correct solution from his prior knowledge. Since the learner wanted to generate a particular example, a demonstration of the correct form of his negative example should be helpful. Such correction is accomplished by analyzing the incorrect input in terms of the templates which have previously been constructed in the cognitive model (student model). Parentheses and quotes are deleted or added where needed, with the following restriction: If a symbol is identified as being a function name, the correction is performed so that the function name yields a function call. Shown below are some examples:

Incorrect:

1. (FIRST (FIRST '(A (B))))
2. (FIRST (REST '(A B)) (REST C D))
3. (LIST (FIRST '(A B) (REST '(C D))))
4. I AM HERE

corrected:

1. (FIRST (FIRST '((A) (B))))
2. (FIRST (REST '(A B (REST C D))))
3. (LIST (FIRST '(AB)) (REST '(C D)))
4. '(I AM HERE).

The corrected examples provide a solution to some task. Supposedly, this solution is identical or at least related to the solution which the learner attempted. We have thus shown, how the cognitive model described at the beginning of the paper can specify the type of information (text or examples) as well as the particular information to be presented to a student for intelligent tutoring.

4. EXPERIMENTAL EVALUATION OF TUTORING SYSTEM

In the following experimental study, three different learning conditions were compared. Learning in the basic exploration environment (with no tutor) was compared to learning with a selective tutor and learning with a constant tutor. Selective and constant tutor differred in their teaching strategy. For the selective tutor the parameters were set to $n = 3$ and $m = 2$. The constant tutor was specified by setting $n = 2$ and $m = 1$. Whereas the constant tutor would provide a correction for every error, the selective tutor would thus only provide a correct example when two errors occurred consecutively. The selective tutor also allows the learner to generate several structurally redundant examples without interfering, whereas the constant tutor provides assistance when a learner has generated two structurally identical examples.

Subjects

Eighteen students from the McGill University/Canada, who were paid $10 for their participation, were randomly assigned to one of the three conditions.

Procedure

After having read an introduction sheet, which outlined the experiment, all subjects acquired knowledge about data representations in LISP (Atoms and Lists) and how the LISP-interpreter evaluates expressions. Thereafter knowledge about the LISP functions FIRST, REST, SET, and LIST was to be acquired under the three different experimental conditions. For each function a single example was presented at the top of the screen. All subjects were instructed to interact with the

exploration environment until they felt that they had acquired sufficient knowledge for solving related programming tasks. Among other data the number of interactions with the exploration environment and their duration were recorded. Upon completion of this learning phase, all subjects were tested with an identical set of 6 programming tasks. The programming tasks required the learners to extract and combine various elements from lists.

Results

The results of the learning phase of the experiment are shown in Table 2: Subjects in the selective tutor condition spent less time in exploring the LISP functions than the subjects in the no-tutor and constant-tutor condition. This time difference can be explained from the fact that the selective tutor subjects generated fewer inputs, and fewer erroneous inputs. Also, a smaller number of tutor corrections was required in the selective tutor conditions. However, none of these group differences were significant.

The results in Table 3 show that the subjects of the tutored conditions required significantly less time for solving the programming tasks than the subjects of the no-tutor condition $F(2,14)=4.6$, MSE=47.3, $p<.05$. All subjects solved about 50% of the programming tasks correctly. So there was no significant difference in the number of correctly solved programming tasks. When the results of the learning and testing phase are considered together, it can be said that the most efficient learning occurred in the selective-tutor condition, followed by the constant-tutor condition. The worst performance was observed in the no-tutor condition.

5. DISCUSSION

In this discussion other tutoring approaches (e.g., Anderson's LISP-tutor) will be compared to the results of the current research. In learning a programming language like LISP, students are normally expected to study a text book and consequently solve a number of related programming problems. Intelligent tutoring systems are typically applied for assisting learners in solving such programming problems (Anderson & Skwarecki, 1986). The tutor determines which set of programming tasks is to be solved. For each programming task a tutor has one or sev-

Table 2. Results of learning phase for the three different conditions

	No Tutor	Selective Tutor	Constant Tutor
Mean Exploration Time: [min]	17.6	11.8	17.8
Mean Number of Inputs:	24.50 (8-59)	17.66 (11-22)	25.33 (5-62)
Mean Number of Erroneous Inputs:	7.16 (1-22)	3.33 (0-6)	7.50 (0-19)
Mean Number of Tutor Corrections:	–	1.00 (0-6)	7.50 (0-19)

Note: In parentheses the respective ranges are shown.

Table 3. Results of the programming tasks for the three different conditions

	No Tutor	Selective Tutor	Constant Tutor
Total Time: [min]	30.54	18.45	18.49
Number of Correct:	2.83	3.00	3.67

eral goal structures, each of which determines a valid solution to the problem. Whenever a learner makes a syntactic error or his programming actions deviate from the tutor's goal structure, the tutoring system immediately provides help information. The learner will therefore always follow one of the tutor's goal structure for solving the task.

Three important assumptions underly this approach: (a) It is better to have the teacher or tutor determine which tasks need to be solved than allowing the student to be creative and make his own selection, (b) all the possible or good solutions to a task can be enumerated and are known beforehand, (c) presumably students cannot learn from their own errors. Therefore students errors should be immediately corrected when they arise. These assumptions may, however, not hold for all learning situations.

Since different learners may prepare themselves to perform quite different tasks with a system, different learners have different learning needs. Learners may also come up with new solutions to a problem, which have not been represented in the tutoring system. Since the errors students make are directly related to their internal goal structure for performing the task, an error message may by itself stimulate useful reasoning processes in the learner. Under some circumstances students may therefore better learn from their own errors than by being presented some correct solution, which is only vaguely related to their own thought processes. Rather than abandoning one's own (erroneous) thoughts completely, by adopting the tutor's correct solution, it may thus be better to straighten out one's own errors and continue the personal line of reasoning.

The experiment presented in this paper demonstrated that students were better in learning from their own errors than having their thought processes be always interrupted by a correct solution. A tutor's advice and corrections are important, however, when the student's misunderstandings are severe. The empirical study showed that the developed computer tutor was rather successful in distinguishing between the student's misunderstandings for which advice and corrections were required and those errors from which learners could recover themselves. In particular, the partial tutor successfully distinguished between these two types of student impasses. The subjects of the partial-tutor condition took the shortest time for learning the relevant functions as well as the fewest inputs to the LISP-interpreter. The partial tutor condition was most effective in that these learners required the least amount of time for acquiring knowledge and solving the criterion test tasks, while solving an equal number of programming tasks correctly. Overall, tutoring was successful, in that the learners in both tutoring condition performed better than in the no-tutor condition. The results thus show that the proposed tutoring strategy was quite successful in combining the advantages of learning by exploration and learning from instrutions.

Acknowledgments — This research was supported by grant Schm 648/1 from DFG to the first author and by an equipment grant from IBM/Canada to McGill University where the tutoring system was initially programmed. The cognitive model was programmed at the Cognitive Science Institute of the University of Colorado, Boulder whose hospitality is greatly appreciated.

REFERENCES

Anderson, J.R., & Skwarecki, E. (1986). The automated tutoring of introductory programming. *Communications of the ACM*, **29**, 842–849.

Dede, C. (1986). A review and synthesis of recent research in intelligent computer-assisted instruction. *International Journal of Man-Machine Studies*, **24**, 329–353.

Hall, R.J. (1988). Learning by failing to explain: Using partial explanations to learn in incomplete or intractable domains. *Machine Learning*, **3**, 1988.

Kühn, O., & Schmalhofer, F. (1987). Erlernen der Computerbenutzung: durch gezielt sequenzierte Instruktion oder durch Explorieren? In W. Schönpflug, & M. Wittstock (Eds.) *Software ergonomie '87*, pp. 387–397, Teubner Verlag, Stuttgart.

Lewis, C. (1988). Why and how to learn why: analysis-based generalization of procedures. *Cognitive Science*, **12**, 211–256.

Mitchell, T.M., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, **1**, 47–80.

Mostow, D.J. (1983). Machine transformation of advice into a heuristic search procedure. In R. Michalski, J.G. Carbonall, & T.M. Mitchell (Eds.). *Machine Learning* (pp. 367–403). Palo Alto, CA: Tioga.

Schmalhofer, F. (1982). *Comprehension of a technical text as a function of expertise.* unpublished doctoral dissertation, University of Colorado.

Schmalhofer, F. (1986). The construction of programming knowledge from system explorations and explanatory text: a cognitive model. In C.R. Rollinger & W. Horn (Eds.), *GWAI-86 and 2nd Austrian Artificial Intelligence Conference*, pp. 152–163. Heidelberg: Springer.

Schmalhofer, F., & Boschert, St. (1988). Differences in verbalization during acquisition from texts, and discovery learning from example situations. *Text*, **8** (4), 369–393.

Schmalhofer, F., & Kühn, O. (1988). Acquiring computer skills by exploration versus demonstration. *The tenth annual conference of the cognitive science society*, Aug. 17–19, Montreal, Canada, 1988, 724–730.

Schmalhofer, F., Kühn, O., & Messamer, P. (1989). Receptive and exploratory learning in intelligent tutoring systems. *Proceedings of the Fourth Annual Rocky Mountain Conference on Artificial Intelligence*, Denver, USA, 1989, 71–82.

Slamecka, N.J., & Kaisaiti, L.T. (1987). The generation effect as an artifact of selective displaced rehearsal. *Journal of Memory and Language*, **26**, 589–607.

van Dijk, T.A., & Kintsch, W. (1983). *Strategies of discourse comprehension.* New York: Academic Press.

Winston, P.H. (1975). Learning structural descriptions from examples. In P.H. Winston (Ed.), *The psychology of computer vision* (pp. 157–209). New York: McGraw-Hill.